Part5

Set 10

The source code for the AbstractGrid class is in Appendix D.

1. Where is the isValid method specified? Which classes provide an implementation of this method?

Whether the location is in the grid ?,

The isValid method is specified in the Grid interface, the BoundedGrid and UnboundedGrid provide an implementations of this method.

2. Which AbstractGrid methods call the isValid method? Why don't the other methods need to call it?

The getValidAdjacentLocations method calls it, because getEmptyAdjacentLocations and getOccupiedAdjacentLocations calls the getValidAdjacentLocations method, which means that they

indirectly call the isValid method. The getNeighbors calls the getOcuppiedAdjacentLocations, so it also

indirectly calls the isValid method.

3. Which methods of the Grid interface are called in the getNeighbors method? Which classes provide implementations of these methods?

The get() method and getOccupidAdjacentLocations method of Grid interface are called in the getNerghbors. The BoundedGrid and UnboundedGrid provide the implementations the get method,

the AbstractGrid provide the implementation of getOccupiedAdjacentLocations

4. Why must the get method, which returns an object of type E, be used in the getEmptyAdjacentLocations method when this method returns locations, not objects of type E?

Because the get method returns the object of type E at the location in the grid if the location is

valid but null if no object. And the getEmptyAdjacentLocations calls the get method to check the

the location is empty, if the get method returns null, then the location is empty, it adds the location

to the ArrrayList of location.

5. What would be the effect of replacing the constant Location.HALF_RIGHT with Location.RIGHT in the two places where it occurs in the getValidAdjacentLocations method?

The adjacent locations to test is only four, that are the north, south, east, west locations, and return these locations which is valid..

Set 11

The source code for the BoundedGrid class is in Appendix D.

1. What ensures that a grid has at least one valid location?

In the constructor of BoundedGrid class, the row and column of boundedgrid must be greater than

zero, this ensure that a grid has at least one valid location. In the code :

```
 if (rows <= 0)
        throw new IllegalArgumentException("rows <= 0");
    if (cols <= 0)
        throw new IllegalArgumentException("cols <= 0");
```

2. How is the number of columns in the grid determined by the getNumCols method? What assumption about the grid makes this possible?

The length of the occupantArray[0], the getNumCols method returns the length of occupantArray[0]

According to the constructor precondition, numRows() > 0, so BoundedGrid at least has one row and one column.

3. What are the requirements for a Location to be valid in a BoundedGrid?

The row value of a location must be greater than or equals to zero and less than the size of the row

of the BoundedGrid, and the column value of a location must be greater than or equals to zero and less than the size of the column of the BoundedGrid.

In the next four questions, let r = number of rows, c = number of columns, and n = number of occupied locations.

4. What type is returned by the getOccupiedLocations method? What is the time complexity (Big-Oh) for this method?

The ArrayList<location> is return, the time complexity to is O(rc).

5. What type is returned by the get method? What parameter is needed? What is the time complexity (Big-Oh) for this method?

The object type E is return. The parameter is Location loc. The time complexity is O(1).

6. What conditions may cause an exception to be thrown by the put method? What is the time complexity (Big-Oh) for this method?

The location loc is invalid may cause an IllegalArgumentException exception and the put object E is null cause an NullPointerException exception .The time complexity is O(1)

7. What type is returned by the remove method? What happens when an attempt is made to remove an item from an empty location? What is the time complexity (Big-Oh) for this method?

The object type E is return. Since the object of a empty location is null, to remove a item from empty location it return null. The time complexity is O(1).

8. Based on the answers to questions 4, 5, 6, and 7, would you consider this an efficient implementation? Justify your answer.

Yes, the implementation is efficient, since only the getOccupiedLocations method's time complexity is O(rc), other method (get, put, remove) all are O(1). To add or to remove an iterm into or from the BoundedGrid is very fast, the time complexity is constant.

Set 12

The source code for the UnboundedGrid class is in Appendix D.

1. Which method must the Location class implement so that an instance of HashMap can be used for the map? What would be required of the Location class if a TreeMap were used instead? Does Location satisfy these requirements?

The Location class must implements hashCode and equals method. When two location is the same, the value of the hashCode is equal and the equals method returns true. If a TreeMap is used, the compareTo method must be implemented, so the Location has to implements the Compare interface. And the Location satisfy these requirements.

2. Why are the checks for null included in the get, put, and remove methods? Why are no such checks included in the corresponding methods for the BoundedGrid?

Because the data structure of UnboundedGrid is HashMap, null is valid in the Map to be a key. Any no-null location in the UnboundedGrid is valid, for the isValid method always return true. But the null location is invalid in the UnboundedGrid, so in the get, put, reomve methods, it has to check wheter the location is null and throw a exception if it is.

In the BoundedGrid, null location is invalid. If the location is null, then in the isValid metho, it return false. If a location to calls the getRow or getCol method without check itself using isValid, it may cause an exception.

3. What is the average time complexity (Big-Oh) for the three methods: get, put, and remove? What would it be if a TreeMap were used instead of a HashMap?

The average time complexity is O(1). If using the TreeMap, the average time complexity is O(lg n) n is the size of the accupantMap.

4. How would the behavior of this class differ, aside from time complexity, if a TreeMap were used instead of a HashMap?

The order of the occupant in the getOccupantLocations may be different sometimes.

Since keys is in the hash table when using HashMap by calculating hashCode with hash function.

Thus order of the key when traverses the keyset depends on the location in the hash table. While the keys is in the balance binary tree when using TreeMap by inserting key in ascending order, the order of the key when traverses th keyset depends on the traverse order.

5. Could a map implementation be used for a bounded grid? What advantage, if any, would the two-dimensional array implementation that is used by the BoundedGrid class have over a map implementation?

Yes, a map could be used for a bounded grid.
The two-dimensional array only store the item, while the map store the key and the item, so the map require more memory.

Exercises

1. Implement the methods specified by the Grid interface using this data structure. Why is this a more time-efficient implementation than BoundedGrid?

Let r = number of rows, c = number of columns, and n = number of occupied locations.

If a program require a large bounded grid, and it often calls the the getOccupiedLocations() method, the time complexity of call it of this implementation is O(r + n), but the time complexity of the implementation of BoundGrid is O(r * c). So the implementation is more time-efficient than BoundedGrid.

2. Fill in the following chart to compare the expected Big-Oh efficiencies for each implementation of the SparseBoundedGrid.

Let r = number of rows, c = number of columns, and n = number of occupied locations

Methods	SparseGridNode version	LinkedList<OccupantInCol> version	HashMap version	TreeMap version
getNeighbors	O(c)	O(c)	O(1)	O(lg n)
getEmptyAdjacentLocations	O(c)	O(c)	O(1)	O(lg n)
getOccupiedAdjacentLocations	O(c)	O(c)	O(1)	O(lg n)
getOccupiedLocations	O(c + n)	O(c + n)	O(n)	O(n)
get	O(c)	O(c)	O(1)	O(lg n)
put	O(c)	O(c)	O(1)	O(lg n)
remove	O(c)	O(c)	O(1)	O(lg n)

3. Implement the methods specified by the Grid interface using this data structure. What is the Big-Oh efficiency of the get method? What is the efficiency of the put method when the row and column index values are within the current array bounds? What is the efficiency when the array needs to be resized?

The Big-Oh efficiency of the get method is O(1).
The Big-Oh efficiency of the put method is O(1) if the row and column index arewithin the current array.
The Big-Oh efficiency of the put method is O(r * c), when the array needs to be resized. Where r and c are the row and column of array.